The Castro AMR Simulation Code: Current and Future Developments

M. Zingale[1], A. S. Almgren[2], M. Barrios Sazo[1], J. B. Bell[4], K. Eiden[1], A. Harpole[1], M. P. Katz[3], A. J. Nonaka[2], D. E. Willcox[2], and W. Zhang[2]

[1]Department of Physics and Astronomy, Stony Brook University, Stony Brook, NY 11794-3800 USA
[2]Center for Computational Sciences and Engineering, Lawrence Berkeley National Lab, Berkeley, CA 94720 USA
[3]NVIDIA Corporation, 2788 San Tomas Expressway, Santa Clara, CA, 95051 USA

E-mail: michael.zingale@stonybrook.edu

Abstract. We describe recent developments to the Castro astrophysics simulation code, focusing on new features that enable our simulations of X-ray bursts. Two highlights of Castro's ongoing development are the new integration technique to couple hydrodynamics and reactions to high order and GPU offloading. We discuss how these features will help offset some of the computational expense in X-ray burst models.

1. Introduction

The Castro astrophysical simulation code [1] is designed for modeling problems in nuclear astrophysics, with the ability to accurately capture the interplay between hydrodynamics, reactions, gravity, and radiation in stars with complex equations of state. Since Castro was first developed, there have been a number of enhancements to the code base, expanding its applicability to a new range of scientific problems. Throughout this development, Castro has strived to perform well on modern supercomputers, adapting to trends in hardware and programming models, while maintaining portability across architectures.

Castro has been applied to models of Type Ia supernovae, core-collapse supernovae, pair-instability supernovae, exoplanet atmospheres, and most recently X-ray bursts (see, e.g., [2–5] for some example science applications). A common challenge in modeling these events is the range of length and timescales involved. To capture length scales, Castro uses adaptive mesh refinement (AMR), through the AMReX library [6]. This allows us to focus the computational effort on regions where burning or the flow is important. To address the range of timescales involved in astrophysical explosions, we have developed a low Mach number hydrodynamics code, MAESTROeX [7], built on the same framework as Castro, that can model the subsonic convection that often precedes explosive events. Both codes are open source and freely available on github[1].

[1] https://github.com/AMReX-Astro/

The most active development presently focuses on new methods of time integration, with better coupling of physical processes, and GPU performance. In these next sections we discuss these features and their impact on our ongoing X-ray burst simulations.

2. Modeling Reactive Flow

In modeling stellar explosions, it is often the case that the hydrodynamics method and the nuclear reaction network require different timesteps in order to produce stable and accurate models. For the stiff nuclear reactions we encounter in stars, implicit ODE methods are often used together with operator splitting to evolve the nuclear reactions separately from the hydrodynamics. In the limit of small timesteps, the two processes are well-coupled together, but these conditions are not always met in simulations. The recent focus in Castro has been on high-fidelity simulations of reactive flow. In [8] we introduced a new time integration strategy, spectral deferred corrections (SDC), that eliminates the coupling error introduced by commonly used operator splitting techniques (see the discussion in [9] for a graphical illustration of splitting error).

The SDC algorithm used in Castro follows the ideas of [10, 11], and uses low order explicit advection and implicit reaction updates in a correction equation that when applied iteratively achieves high-order time-accuracy. In an operator split implementation, reaction and burning proceed without knowing about the other process. In contrast, the update in the SDC formulation explicitly couples the processes together. In Castro we consider reactions and hydrodynamics, and write our conservative system as:

$$\boldsymbol{\mathcal{U}}_t = \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) + \mathbf{R}(\boldsymbol{\mathcal{U}}) \tag{1}$$

where $\boldsymbol{\mathcal{U}}$ is the vector of conserved quantities, $\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})$ is the advective term (the divergence of the hydrodynamic fluxes along with hydrodynamic sources), and $\mathbf{R}(\boldsymbol{\mathcal{U}})$ are the reactive source terms. The update is done at specific temporal nodes, the number of locations of which are picked to give the desired temporal accuracy. The update from one time node m to $m+1$ appears as:

$$\begin{aligned}\langle \boldsymbol{\mathcal{U}} \rangle^{m+1,(k+1)} = \langle \boldsymbol{\mathcal{U}} \rangle^{m,(k+1)} &+ \delta t_m \left[\langle \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) \rangle^{m,(k+1)} - \langle \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) \rangle^{m,(k)} \right] \\ &+ \delta t_m \left[\langle \mathbf{R}(\boldsymbol{\mathcal{U}}) \rangle^{m+1,(k+1)} - \langle \mathbf{R}(\boldsymbol{\mathcal{U}}) \rangle^{m+1,(k)} \right] \\ &+ \int_{t^m}^{t^{m+1}} dt \left(\langle \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) \rangle^{(k)} + \langle \mathbf{R}(\boldsymbol{\mathcal{U}}) \rangle^{(k)} \right)\end{aligned} \tag{2}$$

We use a finite-volume formalism, so $\langle \boldsymbol{\mathcal{U}} \rangle$ is the spatial average of the conserved state in a zone, $\langle \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) \rangle^m$ is the average advective term in a zone at time t^m, and $\langle \mathbf{R}(\boldsymbol{\mathcal{U}}) \rangle^{m+1}$ the average of the reactive source at time t^{m+1}. The second superscript on each term, (k) or $(k+1)$, is the iteration counter. This update is an implicit equation for the new state, $\langle \boldsymbol{\mathcal{U}} \rangle^{m+1,(k+1)}$. The last term in the update is an integral over the sources constructed from the previous iteration's values at each time node. Each iteration of the SDC method increases the temporal order of accuracy by one, up to the order of accuracy with which the integral is constructed.

Castro implements both a second-order method (using a trapezoid rule for the integral) and fourth-order method in space and time (using a Simpsons rule for the integral and the spatial reconstruction of [12]). We demonstrated that Castro achieves the expected convergence on a wide variety of problems in [8]. At the moment, the method is limited to single levels, but work is underway to extend this methodology to AMR. For problems where burning is important and can dominate the computational expense, we hope this new SDC method will become the preferred integration technique in the future.

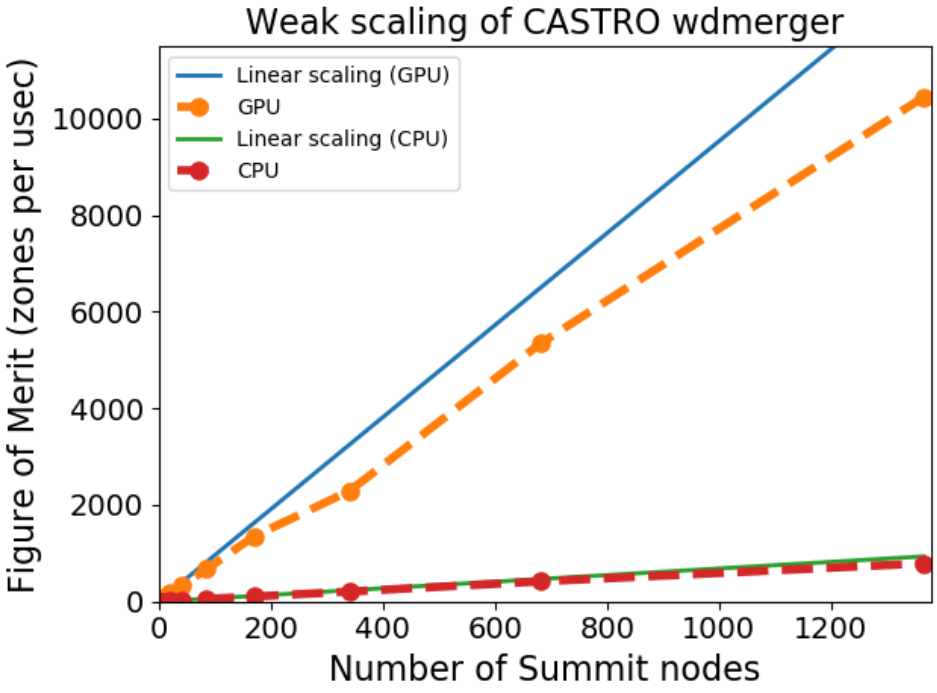

Figure 1. Castro weak scaling on the OLCF Summit machine for the white dwarf merger problem. On each node we use either 42 IBM Power 9 CPU cores or 6 NVIDIA Volta GPUs. Ideal scaling is shown for both cases as the solid line.

3. Performance Portability

The original approach to parallelism in Castro was MPI + OpenMP, with scaling on manycore architectures achieved using a tiling approach to OpenMP [13, 14]. The general approach is to have MPI distribute the AMR grids across nodes and have OpenMP threads work on regions of each grid (tiles) by passing the box describing the tile into the computational kernel. More recently we've ported Castro to GPUs using CUDA, using the same computational kernels as the MPI + OpenMP version. When run on GPUs, each CUDA thread handles the update of a single zone, simply by passing that zone index into the computational kernel. This reuse keeps the code base manageable—we don't need separate kernels for each architecture—while allowing us to take advantage of current and next generation supercomputers. All of the solvers needed to run our core science problems run on GPUs: the main unsplit PPM hydrodynamics scheme [15, 16], self-gravity via multigrid with isolated boundary conditions, thermal diffusion, and nuclear reactions. Our approach is to put the data on GPUs and then run all of the kernels on the GPUs, minimizing data movement. This has given us enormous speed-ups. Figure 1 shows weak scaling on the OLCF Summit machine for a WD merger problem [5]. When using CPUs, we use 42 IBM Power 9 cores per node, while when using GPUs we use 6 NVIDIA Volta GPUs per node. We see that the code performance is about 10× higher per node when using GPUs, while both show excellent weak scaling.

4. Example Application: X-ray bursts

Both the new SDC method and GPU offloading are important to our science goals, in particular the problem of modeling a flame spreading across the neutron star during an X-ray burst (XRB). We discussed the computational challenges of modeling these events in [9]. In short: the largest

scale we need to model is the size of the neutron star, or at least several times the scale at which rotation and the lateral pressure gradient balance (the Rossby length). The smallest scale we need to model is dictated by accurately capturing the burning—we are interested in both the energy release and nucleosynthesis. Our first set of two-dimensional simulations ([9], Eiden et al. in preparation) used two physical approximations to make the problem more tractable. First, we used a higher rotation rate than expected in order to reduce the Rossby length, allowing us to model a smaller region of that star. Second, we artificially boosted the speed of the flame to reduce the duration we need to model. Our next set of calculations will relax these approximations.

With the new fourth-order accurate reactive hydro solver, we believe we can capture the dynamics of the spreading flame accurately with one less refinement level. This will allow us to expand to larger domains while keeping the memory demands reasonable. Full science simulations will begin once we port the SDC framework to AMR and generalize the fourth order solver to axisymmetric geometries. Proof-of-concept single level runs are running now, but the lack of AMR makes them very expensive.

The large increase in performance of the code when run on GPUs enables us to do away with the flame boosting. All of the physics solvers needed for the XRB simulations run on the GPUs: the hydrodynamics, explicit diffusion, realistic equation of state and conductivities, and the 13-isotope He burning network we are using. Preliminary estimates show that we can expect a speed-up of at least 5× for the OLCF Summit nodes (using GPUs) over the NERSC Cori Haswell nodes (using CPUs). Work is underway to further optimize the GPU code, especially the reaction networks.

5. Future Developments

Since its inception Castro has undergone steady development to enable new science investigations and take advantage of new supercomputer architectures. The development of SDC, summarized here, continues, with the current focus on enabling our XRB studies. We are also investigating astrophysical detonations with SDC to understand how well the method works with nuclear statistical equilibrium. For the detonation work, we are exploring different quadrature schemes for the integral in Equation 2 that are more amenable to highly-stiff reaction processes. We are also working on extending the SDC integration methodology to adaptive mesh refinement with subcycling in time.

To complement the existing suite of hydrodynamics solver in Castro, an MHD solver is under development and expected to be merged into the main branch soon. We will use the experiences learned with the Castro hydrodynamics solver to port this solver to GPUs and fit into the SDC framework.

Altogether, these developments will allow us to begin our first three-dimensional realistic XRB calculations by the end of the year, with parameter studies to follow. These calculations will help us understand the nucleosynthesis during XRBs and its impact on observables along with the interpretation of lightcurves and neutron star structure.

Acknowledgments

The work at Stony Brook was supported by DOE/Office of Nuclear Physics grant DE-FG02-87ER40317 and contract 7418390 with Lawrence Berkeley National Laboratory as part of the Exascale Compute Project ExaStar collaboration. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The work at LBNL was supported by the DOE Office of Advanced Scientific Computing Research under Contract No, DE-AC02-05CH11231. An award of computer time was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This research

used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Visualizations were done using yt [17]. This research has made use of NASA's Astrophysics Data System Bibliographic Services.

References

[1] Almgren A, Beckner V, Bell J, Day M, Howell L, Joggerst C, Lijewski M, Nonaka A, Singer M and Zingale M 2010 *The Astrophysical Journal* **715** 1221

[2] Dolence J C, Burrows A and Zhang W 2015 *The Astrophysical Journal* **800** 10

[3] Chen K J, Heger A, Woosley S, Almgren A and Whalen D J 2014 *The Astrophysical Journal* **792** 44

[4] Polin A, Nugent P and Kasen D 2019 *The Astrophysical Journal* **873** 84

[5] Katz M P, Zingale M, Calder A C, Swesty F D, Almgren A S and Zhang W 2016 *The Astrophysical Journal* **819** 94

[6] Zhang W *et al.* 2019 *Journal of Open Source Software* **4** 1370

[7] Fan D, Nonaka A, Almgren A S, Harpole A and Zingale M 2019 *arXiv preprint arXiv:1908.03634*

[8] Zingale M, Katz M P, Bell J B, Minion M L, Nonaka A J and Zhang W 2019 *arXiv e-prints* arXiv:1908.03661 (*Preprint* `1908.03661`)

[9] Zingale M, Eiden K, Cavecchi Y, Harpole A, Bell J, Chang M, Hawke I, Katz M, Malone C, Nonaka A *et al.* 2019 *Journal of Physics: Conference Series* vol 1225 (IOP Publishing) p 012005

[10] Dutt A, Greengard L and Rokhlin V 2000 *BIT Numerical Mathematics* **40** 241–266

[11] Minion M L *et al.* 2003 *Communications in Mathematical Sciences* **1** 471–500

[12] McCorquodale P and Colella P 2011 *Communications in Applied Mathematics and Computational Science* **6** 1–25

[13] Zhang W, Almgren A, Day M, Nguyen T, Shalf J and Unat D 2016 *SIAM Journal on Scientific Computing* **38** S156–S172

[14] Zingale M, Almgren A, Sazo M B, Beckner V, Bell J, Friesen B, Jacobs A, Katz M, Malone C, Nonaka A *et al.* 2018 *Journal of Physics: Conference Series* vol 1031 (IOP Publishing) p 012024

[15] Colella P and Woodward P R 1984 *Journal of Computational Physics* **54** 174–201

[16] Miller G H and Colella P 2002 *Journal of Computational Physics* **183** 26–82

[17] Turk M J, Smith B D, Oishi J S, Skory S, Skillman S W, Abel T and Norman M L 2010 *The Astrophysical Journal Supplement Series* **192** 9